Using VRML to Access Manufacturing Data

Sandy Ressler* Qiming Wang Scott Bodarky
Charles Sheppard Gregory Seidman

Information Technology Laboratory
National Institute of Standards and Technology

Abstract

This paper describes the use of the Virtual Reality Modeling Language (VRML) in the VIM - Visual Interface to Manufacturing system. The VIM prototype demonstrates the technical feasibility of using a variety of interface techniques for data access. VIM is Web based and contains data for the manufacture of a miter saw. Access to manufacturing data is possible via 2D diagrams, and 3D representations (via VRML) of the saw and manufacturing workstations.

CR Categories and Subject Descriptors: H.5.1 [Information Interfaces and Presentation] Hypertext navigation and maps; I.3.2 [Computer Graphics] Graphics Systems - Distributed/network graphics; I.3.6 [Computer Graphics] Methodology and Techniques - Interaction techniques

Additional Keywords and Phrases: virtual environments, user interfaces, data access, manufacturing environment.

1 INTRODUCTION

Manufacturing data comes in a wide variety of forms. Often information exists only on paper. Illustrated part diagrams and process control sheets are used by process engineers and assembly personnel to perform their tasks. Ideally one would like to be able to ask questions of a physical object. An idealized interface, currently not practical, would be to physically hold a real motor, for example, and press on the copper winding or steel housing and have the computer magically understand what one is pressing. A diagrammatic 3D view of the same motor starts to move us, just a little, towards such an idealized interface.

*National Institute of Standards and Technology,
Bldg. 220 Rm. A216, Gaithersburg MD 20899
email: sressler@nist.gov qwang@nist.gov

The type of people we want to use these interfaces are not highly trained computer scientists, rather the users can consist of shop floor assembly line workers without a particularly high level of technical education. Low skill workers should be able to use the 3D interface being prototyped in this system. This hypothesis is as yet unproven. Manufacturing and Process engineers however should have no trouble as the interface is contained within the context of a familiar Web browser.

The VRML object acts as the interface to the database. Queries are effectively hard coded as URL requests. For example, the fact that the guard part of a miter saw, is part 56, is wired into the VRML representation. The data associated with part 56 is stored in a relational database. The associations between the product (saw) and process (workstation) as well as the specific operations which must be accomplished at any workstation, and quality control activities are stored in the database.

2 BACKGROUND

VIM is a prototype Web based system for the access of manufacturing data. The initial test data is from Black & Decker. The data consists of product data such as part numbers, diagrams and descriptions as well as process information. The process information consists of task descriptions for each of over 20 workstation. A workstation is a physical place along an assembly line where a person performs the required tasks. Parts go in one end of the assembly line and completed miter saws come off the end of the line.

One of our goals was to create an intuitive interface through which a naive user can browse for information about both the products and process. Figure 1 illustrates the users entry point to the system. We wanted to be able to answer questions such as “what workstations require part X?” or “what part is supposed to be on the table in workstation Y?”. Most important we wanted to ask these questions in a visual way.

The product and process data was given to us on paper not in any digital form. The Production & PDM Applications project in MEL provided most of the data. Diagrams were scanned and the assembly line and miter saw itself were constructed by hand using the CAD capabilities of Deneb’s QUEST [7] software. In order to more easily move the geometry from QUEST to VRML, we wrote a converter [21] in an earlier phase of the project.

3 RELATED WORK

While VRML is new, much of the component technologies such as hypertext, multimedia, and computer graphics have been around for quite some time. A good survey of these technologies and how they can be used in a manufacturing environment is presented by Leung, Leung and Hill [11]. The use of advanced user interfaces for training applications specific to manufacturing issues is covered in Bengu's examination oriented more towards courseware types of systems [5]. Some of the practicalities of virtual reality and computer graphics as applied to engineering tasks is addressed in Smith, Grimes and Plant's paper [18]. Closely related to our work, a paper by Andrews and Pichler [1], describes the use of 3D models as full-fledged documents.

VRML is rapidly gaining acceptance as a mechanism to visualize geometric representations of a variety of manufacturing related entities. One of the earliest and most effective demonstrations of VRML for an assembly process was the "Computer Desk" site. In this site a user could view in 3D how the pieces of a desk were supposed to be assembled [17]. One significant limitation was the inability to view the pieces of interest in the context of the rest of a completed assembly. Some other VRML sites specific to manufacturing, ship building and process plant industries also exist on the Web [9, 8].

A key difference between VIM and a typical visualization system such as AVS [4] or Data Explorer [10], is that VIM is primarily intended to enable the user to query for data through the visualization. The visualization is the interface to data. Early pre-VRML work [15] explored the use of 3D environments with Web browsers and the current project extends both the Web and graphical interactions.

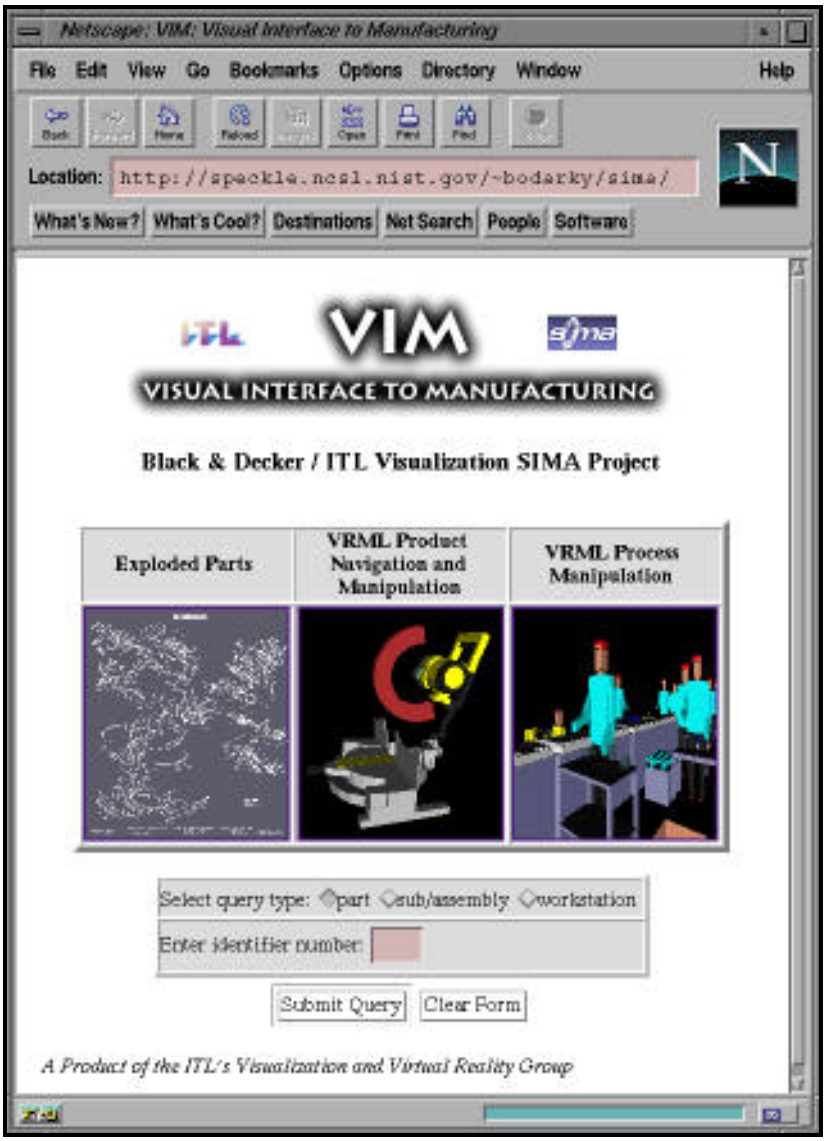

Figure 1. VIM Home page

4 INTEGRATION ARCHITECTURE

The overall architecture, illustrated in Figure 2, is of a typical Web server/client system. A CGI program, the Python script, synthesizes database results and other information into Web pages.

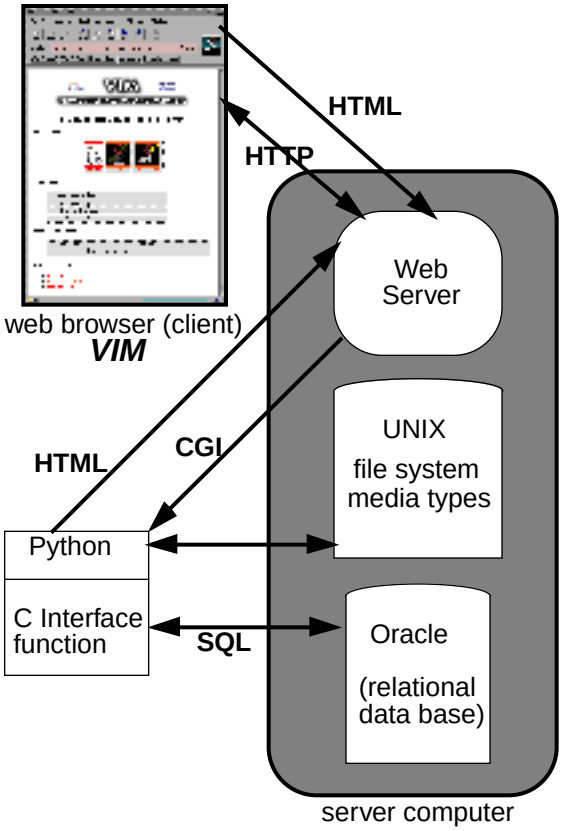

Figure 2. System Architecture

A typical "query" would function as follows:

The user rotates the VRML saw into the desired position and clicks on the particular requested part. The URL contains the call to the Python query script and the part number which is used as data in the C Interface functions. The database is queried for information via the C Oracle interface functions and appropriate database tables are returned. The tables are placed into a memory, via a structure, and a C/Python function turns the data into a form interpretable by Python. The Python scripts examine the data and synthesize the HTML page which is returned to the user.

In addition to using Oracle as the main repository of information associated with parts, process and quality items the UNIX file system was also a media repository. The Web server on which the Python scripts run is also the location for several hundred different files containing a wide variety of multimedia information associated with the parts or workstations. These include digital audio of spoken quality assurance instructions, digital video sequences of workstation operations, QuickTime VR visualizations of some parts, GIF diagrams of the parts, GIF photographs, and Deneb QUEST simulation parameters. The Python scripts display iconic links to these data if they are present on the file system. Figure 3 illustrates these iconic links.

5 WEB GENERATION AND DATABASE

Any Web-based system which interacts a repeatedly with a database must have at least some portion of the Web page generated automatically. In our case we decided to use an interpreted language, Python [19,14] to interpret query results and examine the file system to dynamically generate the resulting Web pages. All of the Web pages containing information for specific parts and workstation were automatically generated, however the introduction and 2D image map page were composed by hand.

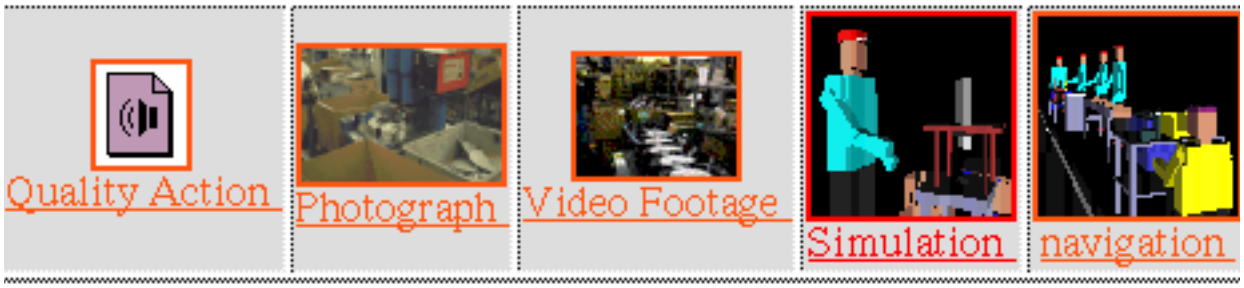

Figure 3. Icons identifying media types present for particular data page.

Figure 4. Workstation report generated from Python script.

An Oracle [13] relational database is used to store the part and workstation information. This included a textual description of the part, the complete part number, operations to be performed at the workstation and so on. The interface between VIM and Oracle is implemented using C code [12] which is wrapped in a Python "module". The actual SQL query to Oracle is accomplished using the C functions provided by the C Oracle API. The tables returned by the query are placed into a C structure which are interpreted by the Python-C code which is in turn called by the Python CGI script. Figure 4 illustrates the Web page constructed by the script.

6 CREATING AND USING VRML

The creation of the miter saw and workstations was accomplished in a two step fashion. First, the objects and workstations were modeled using Deneb's Quest simulation system and the CAD portion of the system. QUEST is a high-end graphics oriented factory simulation system. A custom translator to convert Deneb geometry into VRML was created, because none existed at the time. This translator completely converts Deneb part and workstation geometry into VRML. Kinematic and simulation information are not converted and are a future task.

There are two ways to use the VRML models. One for querying the database called "navigation" and the second for a type of visualization called "manipulation." In order to actually use the VRML models both for querying and visualization VRML manipulation programs were required. For example we want to visualize a particular part rendered solid with the rest of the saw rendered in wireframe. Figure 5 illustrates this type of visualization. This is useful when, a user is viewing the database output for a particular part and prefers to view a 3D model of the saw highlighting that part.

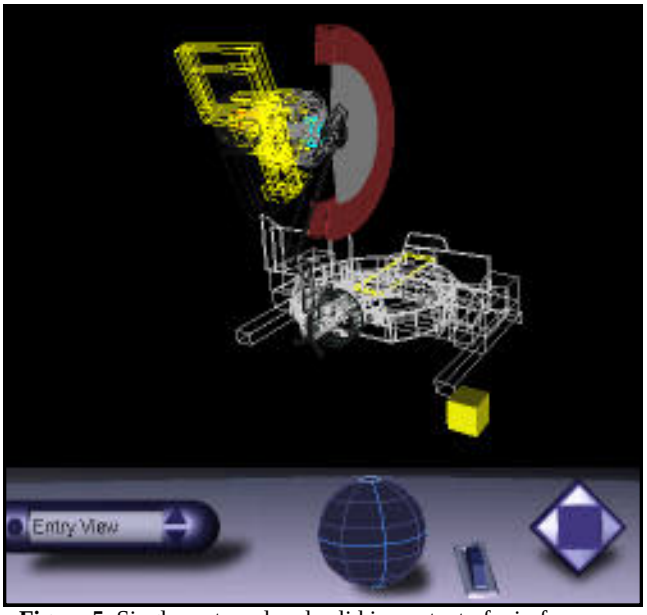

Figure 5. Single part rendered solid in context of wireframe saw.

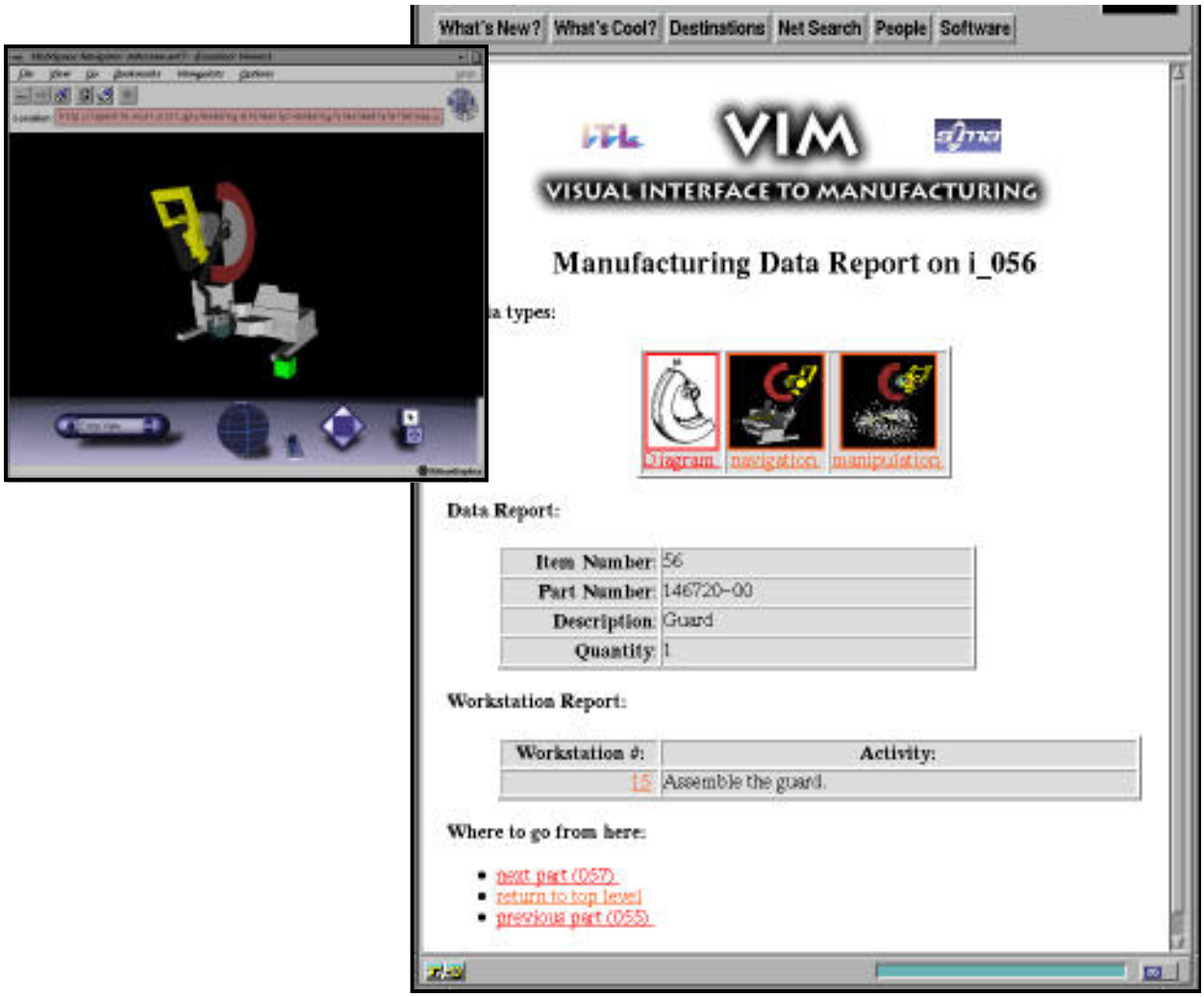

Figure 6. VRML selection window and part report.

We developed a CGI filter program which takes as an argument a specification for which pieces of the saw should be displayed as wireframe and which displayed with solid rendering. The URL embedded in the part information page has this parameter, which is obtained from the database. An additional visualization aid was also developed to allow the user to turn any part of the saw solid or wireframe. Using the same VRML file for both visualization and database selection, however, introduced a problem. The user had no way of controlling whether the part being selected was for a query or for a visualization manipulation. This forced us into the unsatisfying position of creating "modes." A more desirable user interaction, for example using other mouse buttons is anticipated in the future.Mode selection was accomplished by placing a small colored cube in the lower right corner of the saw where the color of the cube tells the user the current mode. A green cube means the system is in query mode and a yellow cube indicates that the system is in manipulation mode. Figure 6 illustrates a VRML selection with the resulting part data report. The two VRML modes, toggling part rendering, and part navigation were implemented using the `manip` filter described below.

In addition to part representations, we created VRML worlds of the assembly workstations. The assembly workstations were modeled using Deneb's QUEST software and translated into VRML via a custom translator. The assembly workstation worlds, illustrated in Figure 7, have links to parts used at the workstation again making these worlds front-ends to database information. One, as yet unimplemented, feature which addresses some of the mode switching problems discussed earlier, is to use the VRML level-of-detail (LOD) feature to enable part and equipment selections and visualizations dependent on the viewer's distance from equipment.

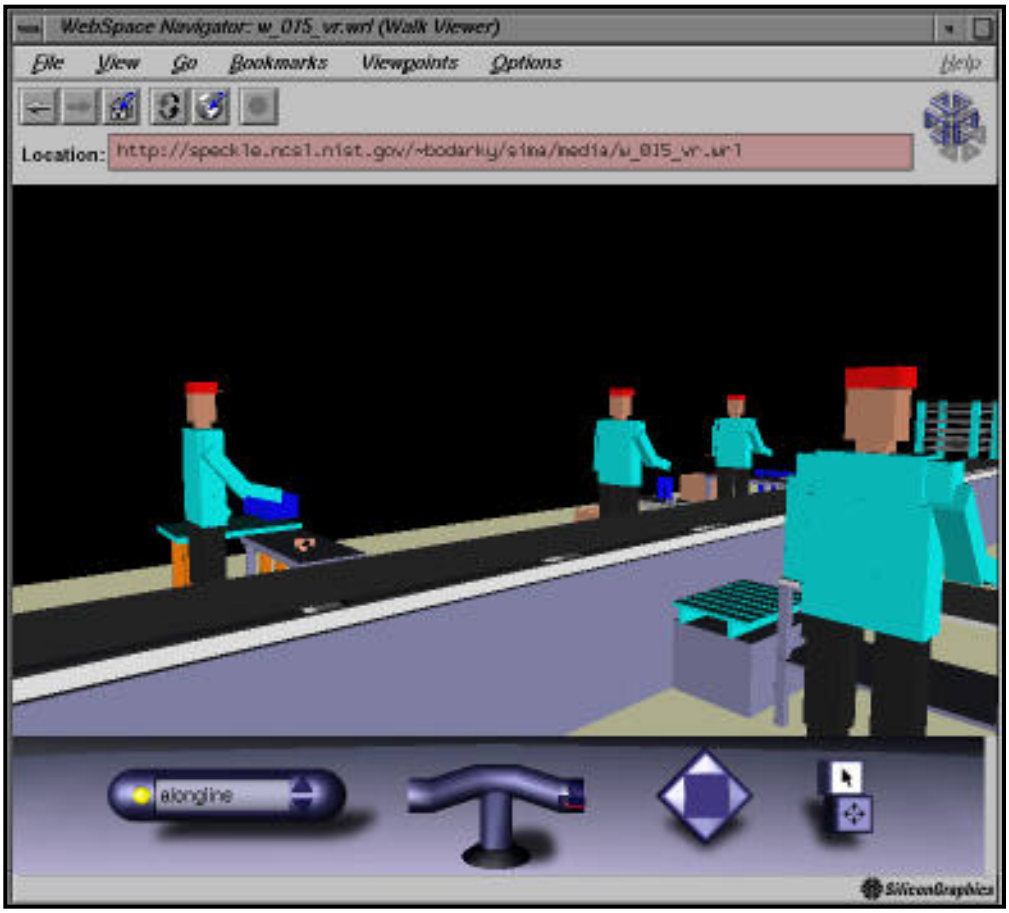

Figure 7. Assembly workstation in VRML, modeled then translated from Deneb's QUEST.

7 MANIPULATING VRML FILES

The manip program was created to allow flexible viewing of individual parts of a larger assembly. Manip is a filter, which takes as input a VRML file and generates a modified VRML file. The modifications are based on parameters passed to manip when it is called. Manip is called as a CGI program and the parameters are passed as CGI parameters. Manip produces VRML files which allow the user to perform two functions. First, it allows the user to selects portions of the VRML file to toggle between wireframe and shaded rendering (manipulation mode), and second, it allow the user to toggle between manipulation and navigation mode. For example, a URL instructing the Web server to execute manip might look like:

```
http://www.my.server/cgi-bin/manip/worlds/
saw.wrl?+1113333111
```

The arguments are the VRML file "/worlds/saw.wrl" and the "+1113333111". The first argument is simply the VRML file. The second argument can begin with either a "+" or a "-" indicating that manip should produce a "manipulation" or "navigation" VRML file.

Manip functions by interpreting special markup codes embedded in a VRML1.0 file. Manip performs two independent operations: First, it tells the VRML browser to render selected parts either wireframe or solid. Second, it hides the URLs used for "navigation" mode. These codes are hidden from VRML interpreters by their placement in a VRML comment, that is, after a # character.

A section of the VRML file is composed of the parts that can be toggled. Each part is surrounded by either a `Separator` node or a `WWWAnchor` node, and there should be no `WWWAnchor` nodes within the part. To use a `Separator` node, `DEF` the `Separator` as `trans_here`, e.g.

```
Separator {
#some VRML here
        DEF trans_here Separator {
                #VRML part
        }
}
```

Using a `WWWAnchor` node is somewhat more complicated, but it should also have a `DEF trans_here`, like this:

```
Separator {
#some VRML here
DEF trans_here WWWAnchor {
        name "http://here.there.com/"
#^%} Separator {
                #VRML part
        }
}
```

Manip also toggles statements in the selected parts from "IndexedFaceSet" to "IndexedLineSet" to actually accomplish the rendering switch between wireframe and solid. To accomplish "manipulation / navigation" mode switching the URLs associated for "navigation" are dissociated from the geometry by manipulating the location of Separators. The separators are identified via a "#^%" character sequence. For additional details on manip see the cited URL [16].

8 MULTIMEDIA / INTEGRATION ISSUES

Multiple media types gives an interface a great deal of flexibility, but it introduces problems unique to each media type. Most problematic of all is the support or lack thereof of a media type during runtime on one or another particular platform. For example the QuickTime VR [3] views of some of the parts are only usable on PCs and Macintoshs. Since most of our work was accomplished on UNIX workstations (SGIs and Suns), the QTVR views did not get much visibility even though they are extremely effective at illustrating real parts and environments. The digital audio sequences had to be converted into `.au` type files which are supported on all the platforms. The digital video (QuickTime) [2] sequences could only be played back on platforms that had all the proper video codecs installed and were capable of sound. QuickTime technically was supported on all of our platforms although some had severe configuration problems. The VRML browser for our development system was WebSpace on an SGI. Other browsers should work just fine.

Additional functionality was created by turning the QUEST simulation package into a Web helper application. This was accomplished by creating QUEST Batch Control Language (BCL) scripts which are interpreted by QUEST. A new MIME [6] type was specified and the Web browser was configured to use that MIME type for files with a `.bcl` extension. While this was a solution for the machines which were licensed to run QUEST, again, this proved to be of no value for other hosts. Lack of simulation capability provides more impetus for the move towards VRML 2.0 versions of the system.

The bottom line is the use of multimedia elements to enhance the user interface can be effective but requires a great deal of attention to the system configuration. In addition, media types such as digital video require a great deal of storage and are unrealistic without video streaming capabilities for low bandwidth (modem) users. These are issues beyond the scope of our project and left to others.

9 CONCLUSIONS AND FUTURE WORK

Although focusing on the VRML interaction this prototype is also a systems integration project. The combination of the infrastructure provided by the Web and the visualization capabilities of VRML offer a unique opportunity to create intuitive 3D front ends to data. The advent of support for behaviors in VRML2 [20] opens the opportunity to move the kinematic and simulation portions of our QUEST simulation into native VRML2. This offers the clear advantage of much wider use, however the production of a complete simulation facility inside VRML2 is a non-trivial task and as yet unknown task.

We intend to move the kinematic and simulation capabilities into VRML2 not to produce a replacement for QUEST, but rather to complement its capabilities. QUEST should be viewed as a potential VRML2 authoring system, provided the proper translators and VRML enhancements are created.

10 ACKNOWLEDGEMENTS

We would like to thank the many people at Black & Decker, particularly Don Elson and Mike Weaver. Thanks also to the Chuck McLean, Swee Leong, Simon Frechette, and Ram Sriram of the Manufacturing Systems Integration Division at NIST for supplying data, photographs and video. Thanks also to Christine Piatko and Sharon Laskowski for their thoughtful reviews of this paper. Finally thanks to the SIMA project management for their continued support.

References

[1] K. Andrews and M. Pichler. Hooking up 3-space: three-dimensional models as fully-fledged hypermedia documents. In *Multimedia, Hypermedia, and Virtual Reality. Models, Systems, and Applications. First International Conference, MHVR '94*, pages 28–44, 1994.

[2] Apple Inc. QuickTime. *http://quicktime.apple.com*, 1996.

[3] Apple Inc. QuickTime VR. *http://qtvr.quicktime.apple.com*, 1996.

[4] AVS. Advanced visual systems. *http://www.avs.com*, 1996.

[5] G. Bengu. Interactive multimedia courseware on manufacturing processes and systems. *International Journal of Engineering Education*, 11(1):46–57, 1995.

[6] Nathaniel Borenstein. Internet multimedia mail with mime: Emerging standards for interoperability. In *ULPAA '92*, Vancouver, 1992.

[7] Deneb Robotics Inc., Auburn, MI. *QUEST User Manuals V2.1*, 1995.

[8] GCRMTC. Gulf coast region maritime technology center. *http://www.luorc.edu/sbd-gcrmtc/*, 1996.

[9] IAI. Information assets, inc. engineering vrml site. *http://www.infoassets.com/vrml/*, 1996.

[10] IBM. IBM visualization data explorer. *http://eagle.almaden.ibm.com/dx/*, 1996.

[11] Ruth F Leung, Horris C Leung, and John F Hill. Multimedia/hypermedia in cim: state-of-the-art review and research implications (part i and ii). *Computer Integrated Manufacturing Systems*, 8(4):255–268, 1996.

[12] Oracle Corporation, Redwood City, CA. *Programmer's Guide to the Oracle Pro*C Precompiler, Release 2.0*, 1993.

[13] Oracle Corporation, Redwood City, CA. *Oracle 7 Manuals*, 1995.

[14] Python. Python language home page. *http://www.python.org/*, 1996.

[15] Sandy Ressler. Approaches using virtual environments with mosaic. In *The Second International WWW Conference '94 Mosaic and the Web*, volume 2, pages 853–860, 1994.

[16] Gregory Seidman. Manip VRML manipulator. *http://www.nist.gov/itl/div878/ovrt/projects/vrml/manip.html, 1996.*

[17] Silicon Graphics Inc. Silicon Depot: CT0405 computer work station desk. *http://webspace.sgi.com/Repository/SGI-Depot/index.html*, 1995.

[18] J.R. Smith, R.V. Grimes, and T.A. Plant. Engineering applications of virtual reality. In *Proceedings SPIE - International Society for Optical Engineering*, volume 2653, pages 332–339, 1996.

[19] Guido van Rossum and Jelke de Boer. Linking a stub generator (ail) to a prototyping language (python). In *Spring 1991 EurOpen Conference Proceedings (May 20-24)*, Tromso, Norway, 1991.

[20] VRML. *VRML 2.0 Specification ISO/IEC CD 14772*, 1996.

[21] Qiming Wang. Deneb2VRML Translator. *http://www.nist.gov/itl/div878/ovrt/projects/vrml/deneb2vrml.html, 1996.*